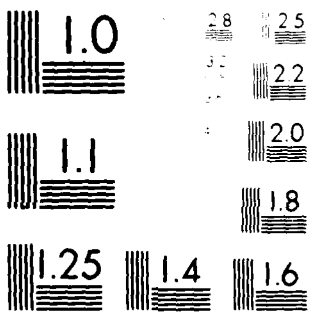
AD-A116 084 RESEARCH INST OF COLORADO FORT COLLINS F/G 20/4
NUMERICAL SOLUTION OF THE THREE-DIMENSIONAL NAVIER-STOKES EQUAT--ETC(U)
MAR 82 J W THOMAS F49620-80-C-0075
UNCLASSIFIED RIC-64 AFOSR-TR-82-0486 NL

1 OF 1
AD A
116084

END
DATE
FILMED
07-82
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

RIC No. 64

6

Final

NUMERICAL SOLUTION OF THE THREE-DIMENSIONAL NAVIER-STOKES EQUATION

F49620-80-C-0075

James W. Thomas
Research Institute of Colorado
Drake Creekside Two, Suite 200
2625 Redwing Road
Fort Collins CO 80526

ABSTRACT

A three-dimensional version of the Beam-Warming scheme for solving the compressible Navier-Stokes equations was implemented on the Cray-1 computer. The scheme is implicit and second-order accurate. The code is totally vectorized, allows for complicated geometries and includes a thin layer turbulence model. Timings and comparisons are given. A preliminary discussion of the full viscous model is included.

I. INTRODUCTION

In recent years substantial progress has been made in the area of numerical simulation of fluid flows. However, one major limitation has been the size and speed of the available computers. Now the new generations of computers, including the ILLIAC IV, the Cray-1, and the CYBER 205, use vector architectures and thus offer new possibilities in the area of fluid flow simulation.

The major purpose of this project was to implement the Beam-Warming scheme for solving the Navier-Stokes equations [1] on the Cray-1. More precisely, the plan was to combine the best parts of the codes written by Steger and Pulliam [2] for the NASA-Ames CDC 7600 and by Lomax and Pulliam [3] for the NASA-Ames ILLIAC IV, so as to take maximum advantage of the Cray-1. Our task was complicated by the fact that the major loops in the two codes were in opposite orders. However, we were able to capture all of the important vector operations of the ILLIAC code and accomplish our goal.

A secondary goal of the project was to include the full viscous effect in the above described code. The implementation of this goal has not been completed.

2. EQUATIONS IN NONDIMENSIONAL FORM

We are interested in the numerical simulation of unsteady, three-dimensional flows of a compressible, viscous fluid in an arbitrary geometry. We wish to use a grid generating scheme so we assume that the geometry of the physical problem given in x-y-z space has been mapped onto a rectangular parallelpiped in the ξ-η-ζ space and that the metrics ξx, ξy, ξz, ηx, ηy, ηz, ζx, ζy, ζz and the Jacobian J of the mapping are provided. (For work on grid generating schemes see [4], [5] or [6].) Hence we must solve the following system of equations.

$$(1)\quad \frac{\partial \Upsilon}{\partial t} + \frac{\partial}{\partial \xi}\left(\hat{\underline{E}} + \hat{\underline{E}}_v - \hat{\underline{E}}_\infty\right) + \frac{\partial}{\partial \eta}\left(\hat{\underline{F}} + \hat{\underline{F}}_v - \hat{\underline{F}}_\infty\right) + \frac{\partial}{\partial \zeta}\left(\hat{\underline{G}} + \hat{\underline{G}}_v - \hat{\underline{G}}_\infty\right) = 0$$

where

$$\underline{q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix} \; ; \; \hat{\underline{E}} = \begin{bmatrix} \rho U \\ \rho uU + \xi_x p \\ \rho vU + \xi_y p \\ \rho wU + \xi_z p \\ (e + p)U - \xi_t p \end{bmatrix} \; ; \; \hat{\underline{E}} = \begin{bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ \rho wV + \eta_z p \\ (e + p)V - \eta_t p \end{bmatrix} \; ;$$

$$\hat{\underline{G}} = \begin{bmatrix} \rho W \\ \rho uW + \zeta_x p \\ \rho vW + \zeta_y p \\ \rho wW + \zeta_z p \\ (e + p)W - \zeta_t p \end{bmatrix} \; ; \; \hat{\underline{E}}_v = J^{-1} \begin{bmatrix} 0 \\ \tau \; \underline{\xi}_m \\ \underline{\beta} \; \underline{\xi}_m \end{bmatrix} ; \; \hat{\underline{F}}_v = J^{-1} \begin{bmatrix} 0 \\ \tau \; \underline{\eta}_m \\ \underline{\beta} \; \underline{\eta}_m \end{bmatrix}$$

$$\hat{\underline{G}}_v = J^{-1} \begin{bmatrix} 0 \\ \tau \; \underline{\zeta}_m \\ \underline{\beta} \; \underline{\zeta}_m \end{bmatrix} \; ; \quad \begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \quad ;$$

$\underline{\xi}_m = [\xi_x \; \xi_y \; \xi_z]^T$; $\underline{\eta}_m = [\eta_x \; \eta_y \; \eta_z]^T$; $\underline{\zeta}_m = [\zeta_x \; \zeta_y \; \zeta_z]^T$;

$\underline{\beta} = \gamma\kappa Pr^{-1} \left[\frac{\partial}{\partial x} \frac{\partial}{\partial y} \frac{\partial}{\partial z} \right] e_I + [u \; v \; w]\tau$; $\quad \tau = [\tau_{ij}]$; $e_I = e\rho^{-1} - .5(u^2 + v^2 + w^2)$

$\tau_{11} = \lambda(u_x + v_y + w_z) + 2\mu \, u_x$; $\tau_{12} = \tau_{21} = \mu(u_y + v_x)$; $\tau_{13} = \tau_{31} = \mu(u_z + w_x)$;

$\tau_{22} = \lambda(u_x + v_y + w_z) + 2\mu u_y$; $\tau_{23} = \mu(v_z + w_y)$; $\tau_{33} = \lambda(u_x + v_y + w_z) + 2\mu w_z$; ρ, u, v, w, e, p are the density, velocity components, internal energy, and pressure ($p = (\gamma-1) \, [e - .5\rho(u^2 + v^2 + w^2)]$), respectively; μ, λ, κ, Pr, Re are the dynamic viscosity, kinematic viscosity, coefficient of thermal conductivity, Prandtl number, and Reynolds number; $\frac{\partial}{\partial x} = (\xi_x \; \eta_x \; \zeta_x) \cdot (\frac{\partial}{\partial \xi} \; \frac{\partial}{\partial \eta} \; \frac{\partial}{\partial \zeta})$, $\frac{\partial}{\partial y} = (\xi_y \; \eta_y \; \zeta_y) \cdot (\frac{\partial}{\partial \xi} \; \frac{\partial}{\partial \eta} \; \frac{\partial}{\partial \zeta})$, $\frac{\partial}{\partial z} = (\xi_z \; \eta_z \; \zeta_z) \cdot (\frac{\partial}{\partial \xi} \; \frac{\partial}{\partial \eta} \; \frac{\partial}{\partial \zeta})$ and $\hat{\underline{E}}_\infty$, $\hat{\underline{F}}_\infty$, and $\hat{\underline{G}}_\infty$ are $\hat{\underline{E}}$, $\hat{\underline{F}}$, and $\hat{\underline{G}}$ evaluated at the free stream values of $\underline{q}$.

3. THIN LAYER APPROXIMATION

For high Reynolds number flows with the assumptions that the η-variables is normal to the mapped rigid boundary and that we are only interested in a thin layer of flow along the rigid surface (or equivalently that the viscous forces away from the surface are negligible), we can eliminate the ξ and ζ derivative terms in the viscous terms ($\underline{\hat{E}}_v$, $\underline{\hat{F}}_v$ and $\underline{\hat{G}}_v$). We then arrive at the following equation.

$$(2)\quad \frac{\partial \underline{q}}{\partial t} + \frac{\partial}{\partial \xi}(\underline{\hat{E}} - \underline{\hat{E}}_\infty) + \frac{\partial}{\partial \eta}(\underline{\hat{F}} - \underline{\hat{F}}_\infty) + \frac{\partial}{\partial \zeta}(\underline{\hat{G}} - \underline{\hat{G}}_\infty) = \frac{1}{Re}\frac{\partial}{\partial \zeta}\underline{\hat{S}}$$

where

$$\underline{\hat{S}} = \begin{bmatrix} 0 \\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)u_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_x \\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)v_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_y \\ \mu(\eta_x^2+\eta_y^2+\eta_z^2)w_\eta + (\frac{\mu}{3})(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta)\eta_z \\ (\eta_x^2+\eta_y^2+\eta_z^2)[.5\mu(u^2+v^2+w^2)_\eta + \kappa Pr^{-1}(\gamma-1)^{-1}(a^2)_\eta] + (\frac{\mu}{3})(\eta_x u + \eta_y v + \eta_z w)(\eta_x u_\eta + \eta_y v_\eta + \eta_z w_\eta) \end{bmatrix}$$

Equation (2) is solved by the Beam-Warming scheme [1] or more specifically the Steger-Pulliam implementation of the Beam-Warming scheme [2]. The technique uses trapezoidal time differencing, expansion of the nonlinear terms about the nth time step, and approximately factoring the resulting equation. We then obtain the following finite difference equation

$$(3)\quad (I + \frac{\Delta t}{2}\delta_\xi A^n - \varepsilon_I J^{-1}\nabla_\xi\Delta_\xi J)(I + \frac{\Delta t}{2}\delta_\eta B^n - \varepsilon_I\nabla_\eta\Delta_\eta J - \frac{\Delta t}{2Re}M^n)(I + \frac{\Delta t}{2}\delta_\zeta C^n - \varepsilon_I \nabla_\zeta\Delta_\zeta J)\Delta\underline{q} = \Delta t(\delta_\xi\underline{\hat{E}}^n + \delta_\eta\underline{\hat{F}}^n + \delta_\zeta\underline{\hat{G}}^n - \frac{1}{Re}\delta_\eta\underline{\hat{S}}^n) - \varepsilon_E J^{-1}[(\nabla_\xi\Delta_\xi)^2 + (\nabla_\eta\Delta_\eta)^2 + (\nabla_\zeta\Delta_\zeta)^2]J\underline{q}^n$$

where Δt is the time increment; δ_ξ, δ_η, δ_ζ are central differences with respect to the appropriate space variable, ε_I and ε_E are amounts of dissipation added (second order and fourth order, respectively); all functions with a superscript m denote that function evaluated at time mΔt; $\Delta\underline{q} = \underline{q}^{n+1} - \underline{q}^n$; and $A = \frac{\partial\underline{\hat{E}}}{\partial\underline{q}}$, $B = \frac{\partial\underline{\hat{F}}}{\partial\underline{q}}$, $C = \frac{\partial\underline{\hat{G}}}{\partial\underline{q}}$ and $M = \frac{\partial\underline{\hat{S}}}{\partial\underline{q}}$. Also the turbulence model described in [7] can be included in Equation (3) by varying the coeficients associated with $\underline{\hat{S}}$. For an excellent paper describing the scheme see [7].

The scheme coded is to solve Equation (3). This process consists of the following four steps: (1) the explicit computation of the right-hand side; (2) - (4) the solution of three block tridiagonal systems of equations.

4. CRAY-1 CODE FOR THIN LAYER APPROXIMATION

The 7600 code to solve Equation (3) is relatively straightforward (or at least as straightforward as such a large code can be). The right-hand side is evaluated and then the three block tridiagonals are solved. An important factor is that the equation to be solved is indexed in the direction associated with that particular factor. For example, when solving the equation

$$(4) \quad (I + \frac{\Delta t}{2}\delta_\eta B^n - \varepsilon_I \nabla_\eta \Delta_\eta J - \frac{\Delta t}{2Re} M^n)\underline{u} = \text{Right-hand side,}$$

it is logical and saves storage space to fix the J and L indices (indices associated with the ξ and ζ directions, respectively) and solve the resulting system of equations indexed by K (index in the η direction). This allows the 7600 code to solve a 30-row (because of size limitations 30 is the largest number of mesh points used in any direction) block (5 x 5 blocks) tridiagonal.

Unfortunately, although the above procedure is probably vectorizable, it is nowhere near optimum for the Cray-1. The largest resulting vector would be 30 where multiples of 64 are the optimum vector sizes on the Cray-1 (and bigger vectors yet gain speed on the CYBER 205).

Because of the vector capabilities of the ILLIAC, the ILLIAC code is quite different from the 7600 code. The ILLIAC code is written in CFD, [8], which is similar to FORTRAN so parts were able to be used almost exactly as in the ILLIAC code. However, due to peculiarities of the ILLIAC, much of the code is ILLIAC dependent (disc manipulations, manipulation of scalars, scalar arithmetic, etc.). Hence, we developed the Cray code by using the serial operation flow of the 7600 while retaining all of the vector portions of the ILLIAC code that are performed often. For another Cray-1 implementation of the ILLIAC code and the subsequent comparison, see [9].

Much of the logic in the ILLIAC code is due to the data structure and manipulation. One problem is that the ILLIAC has only 130K words of memory so to be able to work a very large problem it is necessary to use the disc extensively. Also, two characteristics of the ILLIAC are that it will only handle vectors of length 64 or less (and it is inefficient to use less) and it is very difficult (nearly impossible) to transfer any information down the vector. Because of these limitations the data structure used in the ILLIAC code is to partition the grid into 8 x 8 x 8 blocks. A row of these blocks in a given direction is then called a pencil in that direction. Because of the space limitation of the memory, the scheme is to bring a pencil at a time into core. The solution scheme involves first sweeping through each of the ξ- pencils and with the data in the machine where the η-ζ directions are the components of the vector (a ξ-pencil will contain JMAX 8 x 8 η-ζ planes), calculate the necessary ξ differences in the right-hand side of Equation (3). The next two steps involve doing the same thing with the η and ζ pencils. These steps are necessary with the ILLIAC since the machine will not allow arithmetic along a vector. This implementation is convenient since it makes the right-hand side calculations vector operations. The next three steps involve sweeping through the pencils in each of the directions again, at each step solving the appropriate block tridiagonal matrix. For example, Equation (4) would be solved while sweeping the η-pencils. Since the left-hand side of Equation (4) involves only η differences, the vector contains an 8 x 8 piece of a ξ-ζ plane. This makes solving Equation (4) a perfect vector operation. We might add that for each of these sweeps the data must be in the machine in a different order (to make the operations vector and to make the matrices block tridiagonals). Hence, some of the sweeps also include the appropriate transposes to get the data in the right places. We should also add that by doing the right-hand side calculation and solving the appropriate block tridiagonal matrix during the same sweep for the ξ and η directions, it is necessary to sweep the data four times rather than the six indicated above.

One convenient aspect about converting the ILLIAC code to the Cray was that the above described data structure is also approximately the best to use on the Cray. Initially it might seem that since the Cray vector operations have no length restrictions (except that the speed is optimum when the vector length is a multiple of 64), the best approach is to use pencils that include the entire grid or at least bigger than 8 x 8 planes. This would be possible for problems if a disk version of the Cray code was desired. However, the above scheme for the full grid would require approximately 40 x (grid size) + 50 x (largest plane size) words of storage so only relatively small problems would fit on the Cray.

Our Cray code is written so that it is contained entirely in core. In general, the block tridiagonal solver will require 3 x (vector length) x 25 x (maximum pencil length - 2) words of storage. Allowing space for the full matrix is advantageous because it allows use of canned block tridiagonal solvers such as that described in [10]. We chose not to use this approach. Our block tridiagonal solver generates the matrix during the forward sweep so if 8 x 8 pencils are used, storage due to the matrix solver is minimal (64 x 25 x (pencil length) + 2)).

Thus we have used the same data structure for our Cray code as we used in the ILLIAC code. The storage requirements for our code are approximately 15 x (grid size) + 64 x 43 x (maximum pencil length) + 64 x 90 so at least moderately large problems can be worked using the code.

Hence, the code proceeds to solve Equation (3) by sweeping the data to form the right-hand side and again sweep the data to solve the matrix equations, by the same four sweeps in the ILLIAC code.

5. THIN LAYER CODE RESULTS

It is very difficult and somewhat meaningless to compare results of a code such as this one on different machines. As mentioned earlier the ILLIAC code used a large number of disk manipulations. The 7600 code used the slower large core memory. So to compare times of the Cray code (which is all in core) might not mean much. For lack of other fairer comparisons we will make some of the above comparisons.

The Cray-1 code runs at about 4.9 seconds per time step for a 32 x 32 x 24 grid. The above time includes the starting and ending overhead which we assume is negligible. When the vector mode of the Cray is turned off, the code takes 16.7 seconds per time step for the same grid. Thus it is apparent that the code is gaining efficiency from the vectorization.

It should be noted that in similar tests reported in [9] a Cray code for the Beam-Warming scheme took 1.8 seconds per time step for a 20 x 30 x 21 grid (and 1.3 seconds when using the matrix solver given in [10]). Though this seems like a big difference, it is not. It must be pointed out that a 32 x 32 x 24 grid is almost twice as big as a 20 x 30 x 21 grid. Also the code reported in [9] calculated the metrics associated with the grid generation once and stored them while our code recalculated them for each sweep. Another difference between the two codes is that their code used blocks of size 11, instead of 8. A consequence of their time saving techniques is that largest grid possible for their code without going to disk is 30 x 30 x 38 whereas our code will allow for a 32 x 32 x 48 grid without disk.

An analysis of the time used in the code showing that 40.5% of the time is spent in setting up and solving the block tridiagonal matrices. Thus it might be worth using a canned block tridiagonal solver if the storage requirements make it possible. It might also be worthwhile to develop an efficient block tridiagonal solver that does not require storage of the full tridiagonal matrix.

The best estimates for the ILLIAC and 7600 runs of the analogous material are 5.1 and 19.9 seconds per time step for a 20 x 30 x 21 grid. But again we admit that these comparisons are not really relevant.

It should be noted that the thin layer code was only run with the free stream problem and flow about a hemisphere cylinder. For a discussion of the latter problem see [2]. Since our grids contained of different number of grid points (and it is impossible to use a grid such as theirs in our code), an exact comparison of the two codes was not possible. We did, however, plot the values of p/p_∞ versus x/R as do Steger and Pulliam in [2] (Figure 8) to at least verify that the pressures on the body are the same. The plots were so much the same that we did not reproduce them here.

6. FULL VISCOUS MODEL

As mentioned in the Introduction the inclusion of the full viscous effects has not been completed. There are two reasons why we wish to include the full viscous effects. The first is to see whether they are actually negligible. The second reason is to make it easier to calculate flows in certain complex geometries (for instance, flows in corners). To accomplish the latter goal we felt that it was necessary to include the terms, linearize, factor and then include a three-dimensional turbulence model. After some work this approach was abandoned as being too difficult for the present timeframe.

We next decided that it would be possible to answer the first question sufficiently well by including the remaining viscous (the terms not included in the thin layer approximation) explicitly. We are presently in the process of doing this.

If we return to Equations (1) and (2) it is not hard to see that the parts of (1) that are omitted in (2) are

$$\underline{V} = \frac{\partial}{\partial\xi}\hat{\underline{E}}_v + \frac{\partial}{\partial\zeta}\hat{\underline{G}}_v + \frac{\partial}{\partial\eta} J^{-1} \begin{bmatrix} 0 \\ \tau' \ \underline{\eta}_m \\ \underline{\beta}' \ \underline{\eta}_m \end{bmatrix}$$

where

$$\tau' = [\tau'_{ij}],$$

$$t = \lambda(\xi_x u_\xi + \zeta_x u_\zeta + \xi_y v_\xi + \zeta_y v_\zeta + \xi_z w_\xi + \zeta_z w_\zeta)$$

$$\tau'_{11} = t + 2\mu(\xi_x u_\xi + \zeta_x u_\zeta)$$

$$\tau'_{12} = \tau'_{21} = \mu(\xi_y u_\xi + \zeta_y u_\zeta + \xi_x v_\xi + \zeta_x v_\zeta)$$

$$\tau'_{22} = t + 2\mu(\xi_y v_\xi + \zeta_y v_\zeta)$$

$$\tau'_{23} = \tau'_{32} = \mu(\xi_z v_\xi + \zeta_z v_\zeta + \xi_y w_\xi + \zeta_y w_\zeta)$$

$$\tau'_{33} = \quad + 2\mu(\xi_z w_\xi + \zeta_z w_\zeta)$$

and

$$\underline{\beta}' = \gamma K Pr^{-1} [\xi_x \frac{\partial}{\partial\xi} + \zeta_x \frac{\partial}{\partial\zeta}, \ \xi_y \frac{\partial}{\partial\xi} + \zeta_y \frac{\partial}{\partial\zeta}, \ \xi_z \frac{\partial}{\partial\xi} + \zeta_z \frac{\partial}{\partial\zeta}] \ e_I + [u \ v \ w] \tau'$$

Thus our present plan is to use our present code with the scheme altered so as to solve Equation (3) with a term $\underline{V}^n$ added to the right hand side. If the effects of the full viscous terms are not negligible, then the next step will be to include the terms in $\underline{V}$ implicitly.

REFERENCES

[1] Beam, Richard M. and R. F. Warming, An implicit factored scheme for the compressible Navier-Stokes equations, AIAA 3rd Comp. Fluid Dynamics Conf., Albuquerque, NM, June, 1977.

[2] Pulliam, Thomas H. and Joseph L. Steger, On implicit finite-difference simulations of three-dimensional flow, AIAA 16th Aerospace Science Meeting, Huntsville, Alabama, 1978.

[3] Pulliam, T. H. and H. Lomax, Simulation of three-dimensional compressible viscous flow on the ILLIAC IV computer, presented at the 17th Aerospace Sciences Meeting, New Orleans, LA, Jan. 1979.

[4] Thompson, Joe F., Frank C. Thames, and C. Wayne Mastin, Automatic numerical generation of a body-fitted curvilinear coordinate system for a field containing any number of arbitrary two-dimensional bodies, J. Comp. Physics, 15 (1974), 299-319.

[5] Eisenman, Peter R., A coordinate system for a viscous transonic cascade theory, J. of Comp. Physics 24 (1978), 307-338.

[6] Proceedings of the Symposium on the Numerical Generation of Curvilinear Coordinate Systems and use in the Numerical Solution of Partial Differential Equations, Nashville, Tenn., 1982.

[7] Baldwin, B. S. and H. Lomax, Thin layer approximation and algebraic model for separated turbulent flows, presented at the AIAA 16th Aerospace Sciences Meeting, Huntsville, AL, Jan., 1978.

[8] CFD, A FORTRAN-Based Language for ILLIAC IV, Computational Fluid Dynamics Branch, NASA-Ames Research Center, Moffett Field, Calif., 1974.

[9] Buning, P. G. and J. B. Levy, Vectorization of Implicit Navier-Stokes Codes on the Cray-1 Computer, preprint.

[10] Calahan, D. A., W. G. Ames, and E. J. Sesek, "A Collection of Equation Solving Codes for the Cray-1" SEL Report #133, Systems Engineering Laboratory, Univ. of Mich., 1979.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-0486	2. GOVT ACCESSION NO. AD-A116 684	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NUMERICAL SOLUTION OF THE THREE-DIMENSIONAL NAVIER-STOKES EQUATION		5. TYPE OF REPORT & PERIOD COVERED FINAL, 15 JUN 79-30 SEP 81
		6. PERFORMING ORG. REPORT NUMBER RIC No. 64
7. AUTHOR(s) James W. Thomas		8. CONTRACT OR GRANT NUMBER(s) F49620-80-C-0075
9. PERFORMING ORGANIZATION NAME AND ADDRESS Research Institute of Colorado Drake Creekside Two, Suite 200 2625 Redwing Road, Fort Collins CO 80526		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE MAR 82
		13. NUMBER OF PAGES 11
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Compressible Navier-Stokes equations; beam-warming scheme; vector processor; Cray-1.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A three-dimensional version of the Beam-Warming scheme for solving for compressible Navier-Stokes equations was implemented on the Cray-1 computer. The scheme is implicit and second-order accurate. The code is totally vectorized, allows for complicated geometries and includes a thin-layer turbulence model. Timings and comparisons are given. A preliminary discussion of the full viscous model is included.

7-8